Classificação automática de textos utilizando aprendizado supervisionado baseado em uma única classe

Deni Dias S. Junior, Rafael G. Rossi

[1]Universidade Federal do Mato Grosso do Sul (UFMS)
UNID. II: Av. Ranulpho Marques Leal, 3484 – CEP 79620-080 – Cx Postal nº 210
Mato Grosso do Sul – MS – Brazil

denidias96@gmail.com, rafael.g.rossi@ufms.br

Resumo. *Analisando o volume massivo de textos que trafegam na internet e suas estimativas de crescimento, classificar automaticamente os textos que circulam na rede mundial de computadores tem se tornado interessante para fins acadêmicos e empresariais. Uma das formas de se realizar esta classificação automática é utilizando técnicas de aprendizado de máquina, cujo objetivo é aprender, generalizar, ou ainda extrair padrões ou características de textos. Esses padrões podem ser utilizados, por exemplo, para organizar, classificar e extrair conhecimento de maneira automática. Formas tradicionais de realizar este aprendizado, como o multi-classe, costumam ser utilizados na literatura e em aplicações reais. Entretanto, sua utilização não é adequada quando o usuário deseja extrair conhecimento de apenas uma classe de interesse. Sendo assim, técnicas de aprendizado de máquina utilizando uma única classe, onde há a necessidade de informar apenas documentos da classe de interesse, tem sido pesquisadas nos últimos anos. Tendo em vista esta tendência e sua aplicabilidade em situações práticas, neste trabalho de conclusão de curso serão apresentados métodos de aprendizado de máquina supervisionados baseados em uma única classe para fins de classificação automática de textos, apresentando resultados dos principais algoritmos da área em uma extensa avaliação experimental.*

Introdução

Existem muitos dados no formato digital sendo produzidos e trafegando na rede mundial de computadores. De acordo com estimativas realizadas em 2014, de 2013 a 2020 o universo digital irá aumentar de 4,4 trilhões de gigabytes para 44 trilhões de gigabytes [Turner et al. 2014]. Uma parte destes dados está em formato textual, visto que na internet trafegam dados como e-mails, textos em redes sociais, relatórios e artigos. Uma vez que este é um formato comum para disseminar informações, extrair conhecimento de dados textuais é de interesse empresarial e acadêmico, servindo para analisar novas tendências ou entender o comportamento humano. Contudo, analisando o tráfego existente e sua crescente, torna-se humanamente impossível organizar, gerenciar ou extrair conhecimento destes dados de forma manual. Um meio de automatizar estas tarefas é através da classificação automática de textos [Weiss et al. 2010, Aggarwal and Zhai 2012, Sebastiani 2002].

Técnicas que utilizam a classificação automática de textos tem se tornado cada vez mais estudadas ao longo dos últimos anos [Rossi et al. 2016, Aggarwal and Zhai 2012,

Weiss et al. 2010]. Uma das formas de se realizar a classificação automática de textos é por meio da utilização de técnicas de aprendizado de máquina, cujo objetivo é aprender, generalizar, ou ainda extrair padrões ou características dos textos a serem analisados [Witten et al. 2011, Sebastiani 2002]. O aprendizado se dá pela extração de padrões de coleções textuais apresentadas pelo usuário, fazendo com que seja possível utilizar do seu conteúdo e dos rótulos de seus respectivos documentos para gerar um modelo de classificação, o qual será responsável por classificar documentos que não possuem rótulos.

Uma das formas mais tradicionais para realizar esta classificação utilizando aprendizado de máquina é o multi-classe, também conhecido como *Multi Class Classification*, amplamente utilizado na literatura e em aplicações reais [Witten et al. 2011, Weiss et al. 2010, Sebastiani 2002]. Neste âmbito, o usuário responsável pela rotulação deve definir todas as classes nas quais os documentos poderão ser atribuídos, bem como exemplos de documentos pertencentes a cada uma dessas classes (documentos rotulados). Por exemplo, em um contexto onde serão classificados documentos de notícias, o usuário poderá definir que cada categoria de notícia será uma classe no algoritmo de aprendizado de máquina (e.g. esportes, religião, política), de tal forma que quando for necessário classificar um novo documento, o mesmo será rotulado em uma dessas categorias. Depois de definidas as classes, o usuário então deverá apresentar documentos rotulados de todas as categorias de interesse da aplicação.

Em muitas situações, é de interesse da aplicação ou do usuário que este só rotule textos de seu interesse, visto que no aprendizado multi-classe é necessário rotular textos para todas as classes definidas, mesmo que o usuário não tenha interesse em todas elas. Sendo assim, um documento mesmo que pertença à uma classe não informada no conjunto de treinamento será atribuído à uma das classes presentes no mesmo. Além disso, há situações em que o usuário desconhece todas as possíveis classes de um problema de classificação automática.

Visto que rotular exemplos para todas as classes ainda é um trabalho custoso para o usuário e que o mesmo pode não conhecer todas as classes de um domínio de aplicação, tornou-se então necessária a procura por técnicas que possam solucionar os problemas apresentados pelo aprendizado multi-classe. Uma das técnicas que diminuem o trabalho manual do usuário é o aprendizado de máquina baseado em apenas uma classe, também conhecido como *One Class Classification*. Os estudos baseados neste tipo de aprendizado tem se tornado cada vez mais frequentes nos últimos anos [Liu et al. 2017, Bellinger et al. 2017, Kumar and Ravi 2017, Kemmler et al. 2013]. Esta abordagem permite que o usuário defina uma única classe de interesse e forneça exemplos rotulados pertencentes apenas a esta classe, diminuindo assim o esforço humano. Desta forma, o algoritmo de aprendizado de máquina terá a tarefa de identificar se novos documentos pertencem aquela classe ou não.

Dado o interesse da classificação automática de textos, o desafio do aprendizado baseado em uma única classe e o benefício em aplicações práticas deste aprendizado na classificação automática de textos, o objetivo deste trabalho de conclusão de curso é apresentar estratégias de uso de aprendizado supervisionado baseado em uma única classe para fins de classificação automática de textos, apresentando resultados dos principais algoritmos da área em uma extensa avaliação experimental.

O restante do trabalho de conclusão de curso está dividido da seguinte forma. Na Seção 2 é apresentada a contextualização do presente trabalho. Na Seção 3 são apresentados os trabalhos relacionados a este tema. Na Seção 4 é apresentado o projeto de implementação, descrevendo a avaliação experimental, as coleções textuais utilizadas, a configuração experimental e os resultados deste trabalho. Na Seção 6 são apresentadas as conclusões e os trabalhos futuros.

Conceitos

Uma vez apresentado o objetivo deste trabalho em analisar técnicas para fins de classificação automática de textos utilizando aprendizado supervisionado baseado em uma única classe, serão apresentados os conceitos fundamentais para o entendimento e execução da proposta. Primeiramente serão apresentados conceitos sobre a geração de uma representação estruturada de uma coleção de textos, uma vez que os algoritmos de aprendizado de máquina fazem uso destas representações tanto para aprender quanto para classificar novos textos. Depois são apresentados os conceitos relacionados ao aprendizado de máquina, apresentando as etapas necessárias para sua realização, exemplificando alguns modelos de classificação e descrevendo alguns dos principais algoritmos.

Representação estruturada de coleções

Para realizar a classificação automática de textos, é necessário gerar uma representação estruturada da coleção de textos de tal forma que permita a aplicação comparar os termos contidos entre os textos, extrair padrões e gerar um modelo de classificação, a qual possibilitará a classificação de novos documentos. A forma mais tradicional de se representar textos para classificação é utilizando o modelo espaço-vetorial [Salton 1989], onde cada linha do modelo representa um documento $\mathbf{d}$, e cada coluna representa um termo $\mathbf{t}$, também conhecido como atributo, contidos na coleção de textos. A última coluna é reservada para atribuição de uma classe (rótulo do documento) para os documentos presentes em cada linha da representação. Desta forma, é criada uma matriz documento-termo conforme representado na Tabela 1.

Tabela 1. Exemplo de uma matriz documento-termo representado uma coleção de textos com *m* documentos e *n* termos.

	$\mathbf{t}_0$	$\mathbf{t}_1$	$\mathbf{t}_2$	...	$\mathbf{t}_n$	**Classe**
$\mathbf{d}_0$	$w_{d_0t_0}$	$w_{d_0t_1}$	$w_{d_0t_2}$	...	$w_{d_0t_{n-1}}$	c_{d_0}
$\mathbf{d}_1$	$w_{d_1t_0}$	$w_{d_1t_1}$	$w_{d_1t_2}$	...	$w_{d_1t_{n-1}}$	c_{d_1}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\mathbf{d}_m$	$w_{d_{m-1}t_0}$	$w_{d_{m-1}t_1}$	$w_{d_{m-1}t_2}$	...	$w_{d_{m-1}t_{n-1}}$	$c_{d_{m-1}}$

O valor da célula da matriz documento-termo na linha *i* e coluna *j* ($w_{d_it_j}$), corresponde ao peso do termo t_j no documento d_i. Esse peso pode ser dado, por exemplo, pela frequência de termo no documento (*term-frequency*), pela frequência ponderada pelo inverso do número de documentos (*term-frequency / inverse document frequency*), ou simplesmente pela ocorrência ou ausência de um termo no documento (*binary*).

Uma vez definida a representação a ser utilizada, devemos pré-processar os documentos para se extrair um conjunto de termos conciso, e seus respectivos pesos, na

tentativa de diminuir a dimensionalidade e aumentar a qualidade dos resultados de um processo de mineração de textos. Sendo assim, normalmente as seguintes etapas são realizadas para que possamos classificar de forma mais eficiente as coleções de textos:

1. Padronização dos textos;
2. Remoção de *stopwords*;
3. Radicalização das palavras.

A primeira etapa consiste em padronizar as palavras contidas nas coleções textuais a serem utilizadas, visto que podemos nos deparar com palavras que possuem o mesmo significado, porém, que estão escritas em caixas diferentes, ou com erros de acentuação, gerando diferentes termos para a mesma palavra no momento de representá-las (e.g. Pedágio, pedágio e pedagio). Além disso, podem existir ao longo dos textos caracteres que não serão utilizados para compor os termos, como sinais de pontuação e números. Sendo assim, os seguintes critérios deverão ser utilizados para que as palavras se mantenham padronizadas:

- Retirar acentuação de todas as palavras;
- Retirar pontuação e números;
- Converter todas as letras para minúsculo.

Mesmo depois de ter sido feita esta padronização, é possível verificar dentro dos textos da coleção textual a presença de palavras que não ajudam a discriminar textos de categorias diferentes (e.g. o, as, ele, ela). Estas palavras são chamadas de *stopwords* e mesmo não agregando valor ao assunto abordado no documento, podem influenciar, por exemplo, no momento de calcular a similaridade entre dois textos, visto que podem ocorrer com frequência em ambos [Wilbur and Sirotkin 1992]. A lista de *stopwords*, denominada *stoplist*, geralmente é composta por artigos, pronomes, preposições, advérbios e palavras auxiliares, porém podem variar para cada domínio de aplicação. Portanto, para aprimorar os resultados obtidos pela classificação, são retiradas as *stopwords* presentes dos documentos contidos nas coleções textuais a serem classificadas.

Após realização das etapas anteriores, é possível notar a existência de documentos que utilizam palavras com o mesmo significado, porém, que estão colocadas em tempos verbais diferentes (em caso de verbos), ou que podem variar em gênero, número ou grau (em caso de substantivos), não sendo possível para um algoritmo de aprendizado de máquina equivaler o significado dessas palavras já que estavam representadas em atributos diferentes (e.g. compramos, comprei, compraram). É necessário simplificar estas palavras. A radicalização é uma das formas mais comuns de simplificar as palavras, transformando todas as variações de uma palavra em seu radical comum [Rossi 2016, Dias and de Gomensoro Malheiros 2005]. Sendo assim, todas as variações são representadas em um único atributo radicalizado. Por exemplo, palavras como "problema", "problematizar" e "problematização" serão convertidas para "problem".

Na literatura é possível encontrar outras formas para representar textos, como as representações em redes [Sun and Han 2012]. Dentre estas, destaca-se a rede bipartida, onde são criados objetos para representar os documentos e os termos, onde os pesos são dados pela influência dos termos em cada documento [Rossi et al. 2012].

Aplicada todas as etapas referentes a representação estruturada das coleções textuais, os documentos estarão prontos para serem aplicados em um algoritmo de aprendizado de máquina.

Aprendizado de Máquina

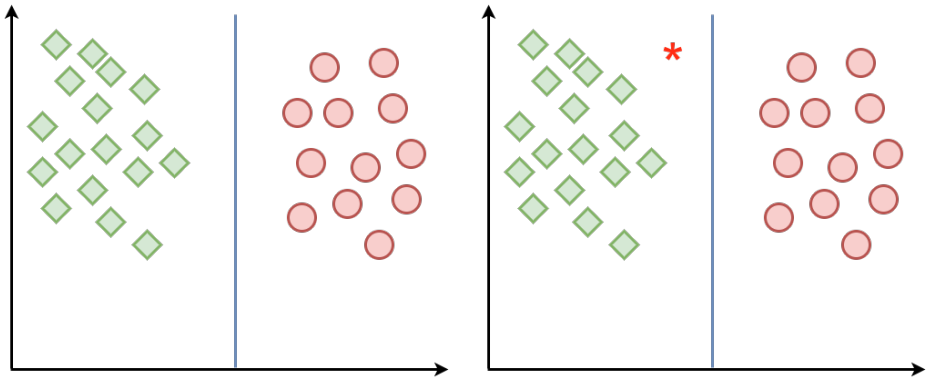
Os algoritmos de aprendizado de máquina, como introduzido na Seção 1, possuem dentro do contexto de classificação automática de textos o objetivo de aprender, generalizar, ou ainda extrair padrões ou características das classes dos textos de uma coleção. Para que seja possível realizar estas tarefas, normalmente representam-se os documentos a serem classificados em um plano *n* dimensional (matriz documento-termo apresentada na seção anterior). A partir desse momento, a grande maioria dos algoritmos de aprendizado de máquina visa aprender uma superfície capaz de dividir os documentos que pertençam a classes diferentes. A superfície de decisão criada pelo algoritmo é chamada de modelo de classificação, isto é, a forma como o algoritmo de aprendizado de máquina irá atribuir um documento às classes pré-estabelecidas. Um exemplo de classificação em um plano utilizando uma superfície de decisão pode ser visualizado na Figura 1. Feito isso, é necessário utilizar uma coleção de documentos já rotulados para que o classificador possa aprender quais são os padrões das classes da respectiva coleção. Estas etapas realizadas até então fazem parte do processo de aprendizado do classificador.

Uma vez passada pela etapa de aprendizado, deverá então ser realizada a classificação, onde usaremos o modelo de classificação definido na etapa anterior para prever a classe de um objeto de interesse. As etapas apresentadas para realizar a classificação automática de textos são genéricas para todos os tipos de classificação utilizando aprendizado de máquina. Um tipo de classificação que é amplamente utilizado na literatura e em aplicações reais é o multi-classe [Witten et al. 2011, Weiss et al. 2010, Sebastiani 2002], onde o classificador possui mais de um rótulo disponível para realizar a classificação, porém somente um rótulo pode ser atribuído por objeto. Sendo assim, em um contexto aplicado a classificação automática de notícias, poderíamos definir um rótulo para cada categoria de interesse (e.g. Esportes, Entretenimento, Economia), porém o algoritmo só poderia rotular uma categoria por notícia.

Diversos algoritmos podem ser utilizados para realizar a classificação multi-classe, tais como *K-Nearest Neighbors* (*kNN*), *Support Vector Machine* (*SVM*) e *Naive Bayes* (*NB*) [Witten et al. 2011, Tan et al. 2006]. Os algoritmos utilizam meios diferentes para induzir um modelo de classificação, isto é, para criar uma superfície de decisão. A seguir são apresentadas algumas formas de criar a superfície de decisão:

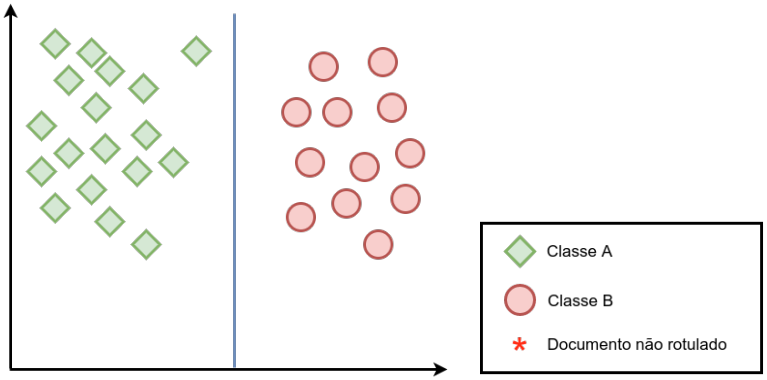
(a) Uma das formas tradicionais de se criar a superfície de decisão é através da utilização de um hiperplano de separação, podendo o mesmo ser perpendicular ao eixo ou não, como representado nas Figuras 2(a) e 2(b). Algoritmos como *SVM* e *NB* fazem uso deste tipo de superfície.
(b) Uma outra maneira de induzir um modelo de classificação é através do uso de superfícies de decisão complexas, que não utilizam formas geométricas definidas para demarcar as classes contidas no plano, como apresentado na Figura 2(c). Este tipo de superfície pode ser obtido, por exemplo, pelo algoritmo kNN.

Apesar de ser o tipo mais utilizado [Sebastiani 2002, Aggarwal and Zhai 2012], a classificação multi-classe apresenta alguns empecílhos que podem acabar dificultando a aplicação prática ou a qualidade do resultado da classificação automática. A primeira dificuldade encontra-se quando o usuário não rotula exemplos para todas as classes do domínio. Neste caso, o classificador gerado por meio do aprendizado multi-classe poderá



(a) Exemplo de uma superfície de decisão extraída de um conjunto de treinamento para as classes ◇ e ◯.

(b) Apresentação de um exemplo não rotulado no plano.



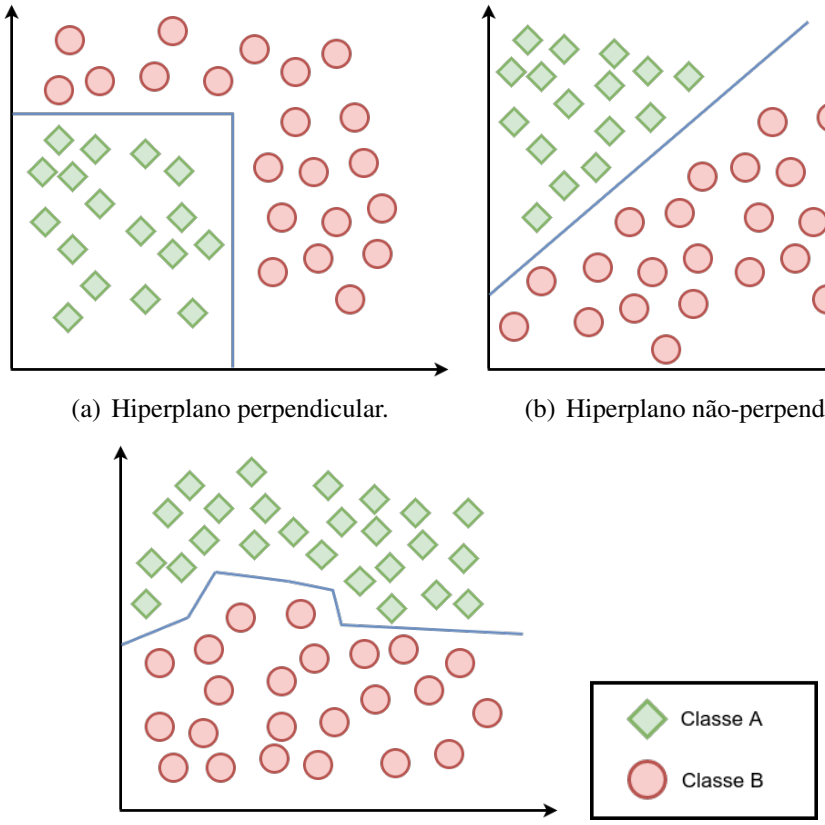
(c) Atribuição do rótulo ◇ ao novo exemplo.

Figura 1. Exemplo uma classificação de um novo documento baseado em uma superfície de decisão de um modelo de classificação.

rotular um novo objeto erroneamente, visto que o mesmo irá forçar a atribuição de um rótulo de uma das classes existentes, como ilustrado na Figura 3.

Semelhante a este problema, há também o caso de o usuário possuir muitos exemplos de documentos de uma ou de algumas classes, e poucos exemplos para o restante das classes. Nesta situação, a coleção de textos estará desbalanceada, dada a desproporcionalidade dos documentos apresentados, fazendo com que o classificador rotule um novo documento tendenciando para as classes que tiveram mais exemplos rotulados. Esta situação é conhecida como problema de desbalanceamento de classes. Um exemplo deste problema utilizando o algoritmo *kNN* para classificar documentos entre as classes ◇ e ◯ é apresentado na Figura 4. Nesta abordagem, o usuário define o número de vizinhos (k) que serão utilizados para classificação, de tal forma que a classe que estiver mais presente entre os k vizinhos mais próximos de um novo exemplo, definirá o rotulo do mesmo. No exemplo ilustrado na Figura 4, para qualquer valor de $k \geq 3$, o exemplo seria atribuído a classe majoritária (◯).

Uma lacuna da classificação multi-classe é quando o usuário possui interesse em apenas uma classe do domínio. Utilizando o exemplo de classificar documentos em um domínio de notícias, mesmo que o usuário tenha interesse apenas na classe que representa



(a) Hiperplano perpendicular. (b) Hiperplano não-perpendicular.

(c) Superfície de decisão complexa.

Figura 2. Exemplos de superfícies de decisão na classificação multi-classe.

notícias de esportes, ele terá que fornecer exemplos de notícias de todas as outras classes para que ele possa obter um resultado minimamente acurado classificador. Isso aumenta o esforço da rotulação por parte do usuário e, portanto, isso pode tornar desinteressante a aplicação da classificação automática em situações práticas.

Outra lacuna da classificação multi-classe é a descoberta de novas classes que não foram definidas pelo usuário, visto que muitas vezes o mesmo pode não ter conhecimento sobre todas as possíveis classes de um domínio de aplicação, ou ainda podem surgir novas classes ao decorrer da aplicação. Sendo assim, o classificador sempre irá classificar um novo objeto apenas entre as classes conhecidas, mesmo que o objeto seja muito diferente de todos os outros já rotulados, como exemplificado na Figura 3.

Analisando estes problemas apresentados pela abordagem de classificação multi-classe, novos métodos para tentar saná-los começaram a surgir, entre eles a classificação automática utilizando uma única classe [Kemmler et al. 2013, Li et al. 2011]. Estudos baseados em aprendizado de máquina utilizando apenas uma única classe, também conhecida como *One Class Classification*, tem se tornado mais frequentes nos últimos anos [Liu et al. 2017, Bellinger et al. 2017, Kumar and Ravi 2017, Kemmler et al. 2013]. Nesta abordagem, o usuário define apenas uma classe de interesse, rotula alguns exemplos desta classe e o algoritmo de aprendizado de máquina identificará se um novo documento pertence à classe de interesse ou não. O aprendizado baseado em uma única classe é mais desafiador que o aprendizado multi-classe, pois no último, ao fornecer exemplos de

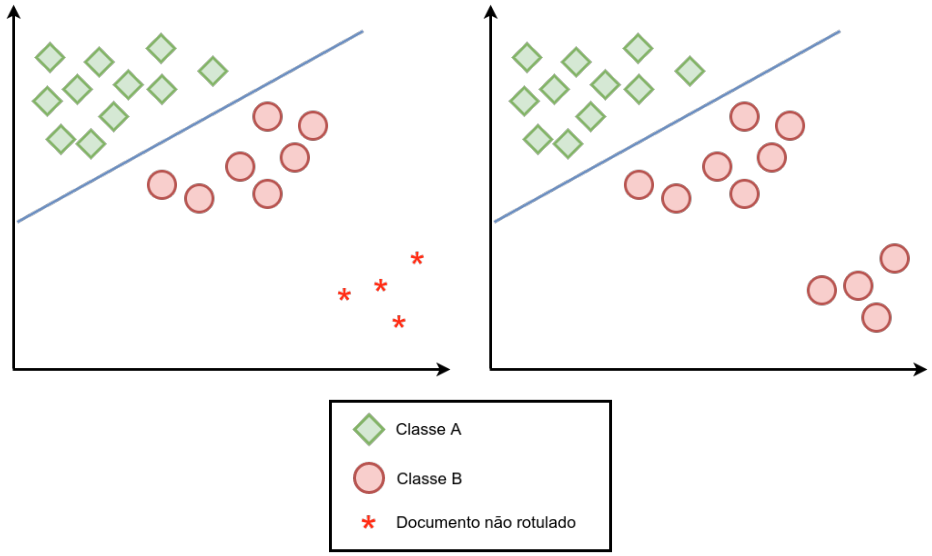
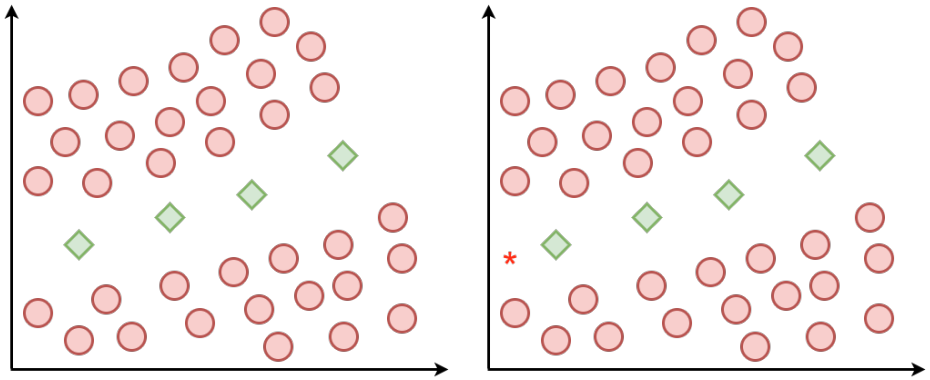


Figura 3. Exemplo de uma classificação do classificador multi-classe quando não há exemplos para todas as classes.

duas classes distintas, pode-se inferir com mais facilidade a superfície de decisão que separa uma classe da outra. Por outro lado, no aprendizado baseado em uma única classe tem que se inferir baseando-se apenas nos exemplos da classe de interesse, uma vez que nem sempre é possível obter exemplos contrários aos da classe, denominados de *outliers*. Esta abordagem facilita no momento de induzir um modelo de classificação utilizando exemplos desbalanceados, obtendo um desempenho melhor que o multi-classe para este fim [Zhuang and Dai 2006]. Na Figura 5 é possível observar modelos de classificação baseados em uma única classe.

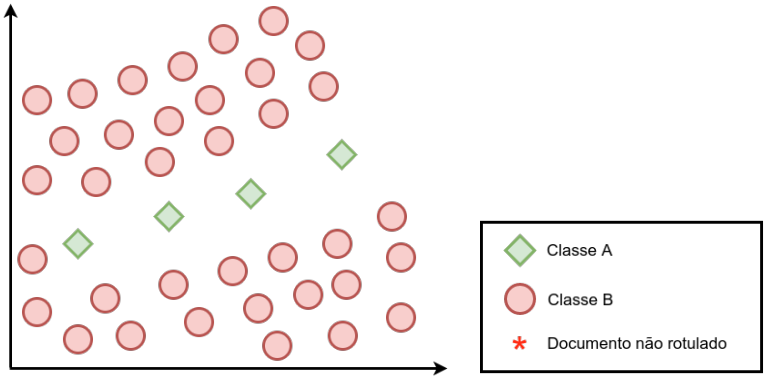
Mesmo com este desbalanceamento, os algoritmos de classificação baseados em uma única classe (*One Class Classification*) se saem melhor que os algoritmos de classificação multi-classe, uma vez que neste último caso o algoritmo estará tendenciado a rotular um novo documento para a classe mais similar, podendo esta não ser a classe definida para capturar *outliers*, como já apresentado na Figura 5 [Zhuang and Dai 2006].

Mesmo com as diferentes formas de se criar uma superfície de decisão, como apresentado na Figura 5, o modo como os algoritmos definem se um novo documento pertence à classe de interesse é comum entre todos eles. Primeiramente é necessário utilizar o modelo de classificação induzido pelo algoritmo de aprendizado de máquina baseado em uma única classe para gerar um *score* para o novo documento a ser classificado, isto é, gerar um valor que irá indicar o quão próximo o documento está de pertencer a classe de interesse ou não. Este modelo de classificação será denotado aqui por $f(\mathbf{d}_i)$. A partir desse momento, o usuário responsável pela aplicação deve definir um limiar, também denotado como *threshold*, que define um valor mínimo necessário para que um novo documento, através de seu *score*, possa ser rotulado como sendo da classe de interesse. Caso o *score* gerado para o novo documento seja maior ou igual ao *threshold*, ele é considerado como pertencente a classe de interesse. Caso ele fique abaixo do *threshold*, ele é considerado um *outlier*. Abaixo é apresentada a função que define se um documento pertence ou não a classe de interesse definida.



(a) Plano onde os documentos estão sendo representados exemplos das classes ◇ e ◯.

(b) Apresentação de um novo exemplo não rotulado no plano.



(c) Atribuição do rótulo ◯ ao novo exemplo.

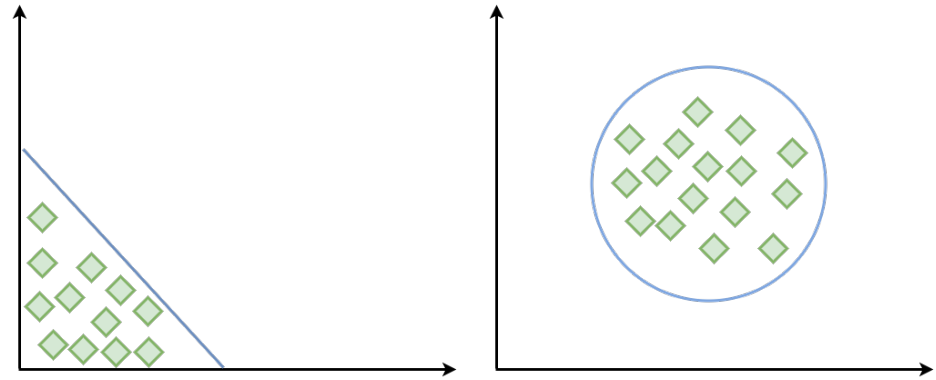
Figura 4. Exemplo de uma classificação equivocada do classificador multi-classe devido a exemplos desbalanceados.

$$classe = \begin{cases} f(\mathbf{d}_i) \geq threshold \rightarrow interesse \\ f(\mathbf{d}_i) < threshold \rightarrow outlier \end{cases} \quad (1)$$

Para realizar a classificação automática de textos baseado em uma única classe são utilizados diversos tipos algoritmos de aprendizado de máquina. A seguir são descritos alguns dos principais algoritmos que realizam este aprendizado e que são utilizados neste trabalho de conclusão de curso [Rossi et al. 2012, Khan and Madden 2009, Tan et al. 2006, Tax 2001].

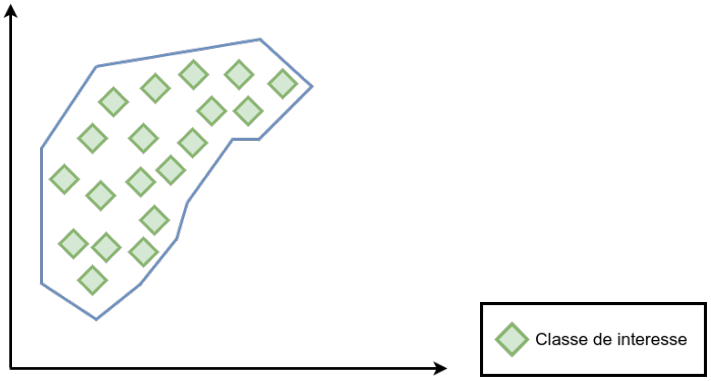
Multinomial Naive Bayes

O algoritmo *Multinomial Naive Bayes* [Aggarwal and Zhai 2012], faz uso de uma distribuição multinomial para gerar a probabilidade de um documento pertencer a classe de interesse. Para avaliar se um novo documento pertence à classe de interesse é necessário então gerar um *score* através da função apresentada abaixo.



(a) Modelo de classificação obtido por meio de um hiperplano.

(b) Modelo de classificação obtido por meio de uma hiperelipse.



(c) Modelo de classificação utilizando superfície de decisão complexa.

Figura 5. Exemplos de superfícies de decisão utilizando aprendizado de máquina baseado em uma única classe.

$$f(\mathbf{d}_i) = \prod_{\mathbf{t}_j \in d_i} P(\mathbf{t}_j) \tag{2}$$

A função $P(\mathbf{t}_j)$ é dado pela probabilidade de ocorrer o termo $\mathbf{t}_j$ em um documento da classe de interesse. A função que calcula esta probabilidade é mostrada abaixo.

$$P(\mathbf{t}_j) = \frac{1 + \sum_{\mathbf{d}_i \in D} w_{d_i t_j}}{|T| + \sum_{\mathbf{t}_k \in T} \sum_{\mathbf{d}_i \in D} w_{d_i t_k}} \tag{3}$$

kNN Density-based

Variação do popular algoritmo *k-Nearest Neighbors*, o *kNN Density-based* [Tan et al. 2006] utiliza a densidade de exemplos próximos a um documento para avaliar se o mesmo pertence à classe de interesse ou não. Desta forma, é possível estimar que um documento pertence à uma classe de interesse se este estiver em uma

região de alta densidade. Para calcular esta densidade, o usuário deve definir o número de *k* vizinhos a serem utilizados. Feito isso, a densidade será dada pela média das proximidades dos *k* vizinhos do documento, isto é:

$$densidade(\mathbf{d}_i, k) = \frac{\sum_{\mathbf{d}_j \in \mathcal{N}_{(\mathbf{d}_i,k)}} proximidade(\mathbf{d}_i, \mathbf{d}_j)}{|\mathcal{N}_{(\mathbf{d}_i,k)}|}, \tag{4}$$

na qual a variável $\mathcal{N}(\mathbf{d}_i, k)$ representa o conjunto de vizinhos mais próximos de $\mathbf{d}_i$. Para calcular a proximidade entre dois documentos, foi utilizada a medida de similaridade Cosseno, indicada para comparar textos uma vez que compara o ângulo formado pelos termos presentes nos documentos, e não necessariamente o quão próximos estão os documentos no espaço euclideano [Salazar 2012]. A função para calcular a proximidade entre um documento $\mathbf{d}_i$ e um documento $\mathbf{d}_j$ utilizando a medida de similaridade Cosseno é apresentada na Equação 5.

$$proximidade(\mathbf{d}_i, \mathbf{d}_j) = \frac{\sum_{\mathbf{t}_k \subset \mathcal{T}} w_{d_i,t_k} \cdot w_{d_j,t_k}}{\sqrt{\sum_{i=0}^{n-1} (\mathbf{d}_i)^2} \cdot \sqrt{\sum_{i=0}^{n-1} (\mathbf{d}_j)^2}} \tag{5}$$

kNN Relative-Density-based

Outro método para classificação utilizando variações do *k-Nearest Neighbors* é o *kNN Relative-Density-based*. Esta técnica compara a proporção entre a densidade do objeto a ser classificado $\mathbf{d}_i$ com a média da densidade de seus vizinhos. Quanto mais parecidas as densidades, mais próximo de um será o valor retornado pela função. A função abaixo mostra como calcular a densidade relativa para um novo documento.

$$densidade\_relativa(x, k) = \frac{densidade(\mathbf{d}_i, k)}{\sum_{\mathbf{d}_j \in N(\mathbf{d}_i,k)} densidade(\mathbf{d}_j, k)/|\mathcal{N}(\mathbf{d}_i, k)|} \tag{6}$$

A função para calcular a densidade, assim como a função para calcular a proximidade baseando-se na medida de similaridade do cosseno estão descritas nas Equações 4 e 5 respectivamente, onde foi apresentado o *kNN Density-based*.

IMBHN-based

Algoritmo proposto em [Rossi et al. 2016] especialmente para classificação automática de textos, o *Inductive Model Based on Bipartite Heterogeneous Network* (*IMBHN*) induz um modelo de classificação por meio da obtenção de um *score* de relevância dos termos

para as classes de uma coleção. Neste projeto, foi proposto uma adaptação do algoritmo *IMBHN*^{R}. Essa adaptação consiste em obter um peso dos termos da coleção de forma que eles sejam capazes de predizer a classe de interesse, neste caso, predizer as classes de interesse significa gerar valores próximos a 1. Abaixo é possível visualizar a função que calcula a frequência entre os termos.

$$F(\mathcal{T}) = \sum_{\mathbf{d}_k \in D} (1 - (\sum_{\mathbf{t}_i \in T} w_{d_k t_i} \cdot F_{t_i})) \tag{7}$$

O *score* de um novo documento $\mathbf{d}_i$ é dado por

$$f(\mathbf{d}_i) = \sum_{\mathbf{t}_j \in \mathcal{T}} w_{d_i t_j} \cdot F_{t_j} \tag{8}$$

kMeans-based

Esta abordagem utiliza o algoritmo *kMeans* para segmentar os exemplos da classe de interesse em *k* grupos, de forma que os documentos em um grupo sejam mais similares entre si e menos similares em relação a documentos de outros grupos [Tan et al. 2006]. Para cada grupo é gerado uma instância fictícia que representa o centro daquele grupo, chamada de centroide. Para definir a qual grupo pertence um determinado exemplo rotulado, é necessário calcular a distância do mesmo para cada um dos centroides dos *k* grupos. O *score* será dado então pela similaridade entre o documento e o centroide mais próximo. A medida utilizada no presente trabalho para calcular a similaridade entre duas instâncias foi a medida de similaridade do cosseno, apresentada na Equação 5.

Trabalhos relacionados

Conforme apresentado na Seção 1, pesquisas relacionadas as técnicas de aprendizado de máquina utilizando apenas uma única classe tem se tornado frequentes nas últimas décadas. No trabalho realizado por [Khan and Madden 2009], as tendências de pesquisas relacionadas a este tipo de classificação estão divididas da seguinte forma:

1. Disponibilidade de dados para treinamento
2. Metodologias usadas para classificação
3. Domínios a serem aplicadas as classificações

Dentro destas três divisões, pesquisas relacionadas a classificação automática de textos foram desenvolvidas com objetivos e resultados distintos.

No trabalho apresentado por [Manevitz and Yousef 2001], foi utilizado o algoritmo *Support Vector Machine* adaptado para o *One Class Classification*, induzindo um modelo de classificação baseado em um hiperplano não perpendicular como superfície de decisão para realizar a separação da classe de interesse em relação aos *outliers*. O experimento foi aplicado apenas em uma coleção textual, a *Reuters*, amplamente utilizada para testes de classificação de textos [Hotho et al. 2003].

No trabalho apresentado por [Liu et al. 2003], foi explorada a classificação de textos baseando-se na disponibilidade apenas de exemplos pertencentes apenas a classe de

interesse e não rotulados, não existindo exemplos rotulados de *outliers*. Neste trabalho, o processo de classificação foi dividido em duas etapas:

1. Utilização de algoritmos de aprendizado máquina para extrair exemplos negativos (*outliers*) entre os exemplos não rotulados.
2. Aplicação iterativa de algoritmos de aprendizado de máquina definidos afim de obter um melhor classificador para cada coleção de textos utilizada. Para obter o melhor classificador foi necessário fazer uma combinação entre os algoritmos da primeira etapa com os utilizados nesta etapa, calcular a média harmônica dos resultados de classificação para então identificar qual combinação se saiu melhor.

No fim, foi apresentado os resultados dos algoritmos para cada coleção textual, e proposto uma adaptação do algoritmo de *Support Vector Machine*, denominado *Biased-SVM*, com o objetivo de utilizar o *Support Vector Machine* para fins de classificação utilizando uma única classe.

No trabalho apresentado por [Peng et al. 2006], também foi possível explorar a classificação de textos baseando-se na indisponibilidade de exemplos negativos. As etapas para realizar a classificação foram semelhantes ao trabalho de [Liu et al. 2003]. Inicialmente foram identificados documentos negativos dentre os documentos não rotulados utilizando algoritmo de aprendizado de máquina. Depois foi utilizado o algoritmo *Support Vector Machine* com múltiplas iterações para realizar a classificação de novos documentos não rotulados. O diferencial deste trabalho é o uso de algoritmo genético para definir qual o melhor número de iterações para executar o *SVM* nos domínios onde a classificação foi aplicada, utilizando a média harmônica dos classificadores como função de *fitness*.

De acordo com o objetivo definido para este trabalho de conclusão de curso, iremos explorar o domínio de textos utilizando apenas da classe de interesse. A proposta apresentada neste trabalho de conclusão de curso, apesar dos trabalhos apresentados nesta seção, ainda não foi explorada na literatura, visto que ainda não foi feita uma extensa avaliação experimental sobre classificação automática de textos utilizando aprendizado supervisionado baseado em uma única classe.

Projeto de implementação

Uma vez apresentada toda a parte conceitual que rege o presente trabalho e os trabalhos relacionados, será apresentado nesta seção o projeto de implementação criado para realizar uma extensa avaliação experimental relacionado ao tema em questão. Primeiramente foi necessário definir em qual linguagem de programação aplicá-los para extrair os resultados da avaliação. A linguagem de programação definida foi Java, tendo em vista a praticidade de se rodar programas gerados pela linguagem em múltiplos sistemas operacionais [Gosling 2000, Gosling and McGilton 1995]. Além disso, a escolha também se baseou em alguns recursos que facilitariam no processo de desenvolvimento dos algoritmos, tais como a utilização de estruturas de dados contidas nas bibliotecas da linguagem (e.g. *hashs* e listas encadeadas), praticidade para implementação de técnicas de programação orientada a objetos (e.g. classes abstratas e interfaces) e uso de paralelismo para execução dos algoritmos.

Além de fazer uso dos recursos da linguagem, foi feito uso a biblioteca da ferramenta *Weka*, amplamente utilizada para realizar experimentos voltados para aprendizado

de máquina [Witten et al. 2011]. Foram aplicados diversos dos recursos da biblioteca no projeto de implementação, entre eles está a facilidade de carregamento de arquivos no formato *ARFF* contendo as matrizes documento-termo que representam as coleções de textos. Além disso, foi necessário utilizar as representações citadas acima para criar conjuntos de treino e teste dos documentos rotulados para execução dos experimentos, que é outra facilidade provida pela biblioteca *Weka*. Outro recurso utiliza que ainda será citado em diante foi a classe para construção de novos classificadores, a *Classifier*.

Toda implementação realizada durante o desenvolvimento deste trabalho de conclusão de curso foi aplicada a ferramenta *Text Categorization Tool*, desenvolvida para automatizar experimentos de classificação automática de textos [Rossi 2016].

Nas próximas subseções são apresentados mais detalhes sobre as implementações realizadas neste trabalho.

One Class Classifier

Para realizar a implementação dos algoritmos foram utilizados recursos da programação orientada a objetos, de forma a facilitar a compreensão do código, modularizar os métodos a serem utilizados e reaproveitar funções da ferramenta *Weka*. Portanto, foi criado dentro da ferramenta *Text Categorization Tool* um pacote somente para os algoritmos de *One Class Classification* que contém a implementação de algoritmos de aprendizado de máquina baseados em uma única classe.

A fim de centralizar o que é padrão entre os algoritmos de *One Class Classification*, foi criada uma classe abstrata denominada *OneClassSupervisedClassifier*, que herda as propriedades da classe *Classifier* do *Weka*. Para construir o modelo de classificação e rotular novos documentos de acordo com as características de cada algoritmo, é necessário sobrepor os seguintes métodos herdados da classe *Classifier*:

(a) `buildClassifier(Instances instancias)`: método utilizado para construir um modelo de classificação a partir de um conjunto de treino passado por parâmetro. A classe *Instances* é responsável por armazenar instâncias no formato do *Weka*;
(b) `classifyInstance(Instance instancia)`: método utilizado para classificar um novo documento não rotulado dado o modelo de classificação induzido na função *buildClassifier*. O documento em questão é passado como parâmetro da função.

Além dos métodos herdados da classe *Classifier*, foram criados dentro da classe *OneClassSupervisedClassifier* os métodos abaixo:

(a) `setThreshold(double threshold)`: método desenvolvido para atribuir um *threshold* ao classificador;
(b) `getThreshold()`: método desenvolvido para recuperar um *threshold* previamente estabelecido. Caso não tenha sido estabelecido nenhum *threshold*, este método retorna um valor nulo;
(c) `getScore(Instance test)`: método abstrato para gerar um *score* para a instância de teste passada por parâmetro.

A partir desse momento, cada algoritmo é implementado estendendo a classe *OneClassSupervisedClassifier*, sobrescrevendo o método *buildClassifier* para aplicar o modelo de classificação proposto pela abordagem do algoritmo, o método *getScore* para atribuir um *score* a um documento de teste e, por fim, o método *classifyInstance* que irá definir o rótulo do documento de teste através do *score* gerado e do *threshold* definido para o algoritmo. O método *classifyInstance* irá retornar 0 caso o documento seja rotulado como sendo da classe de interesse e 1 se for rotulado como *outlier*, facilitando no momento de gerar a matriz de confusão, métrica que será apresentada adiante neste trabalho.

Algoritmos

Após definição da linguagem, foram definidos os algoritmos a serem utilizados na avaliação experimental, apresentados previamente na Seção 2.2. Portanto, os algoritmos selecionados para implementação neste trabalho de conclusão de curso foram:

- *Multinomial Naive Bayes-based*
- *kNN Density-based*
- *kNN Relative-Density-based*
- *IMBHN^R-based*
- *kMeans-based*

O *Multinomial Naive Bayes-based* foi implementado aproveitando os modelos de classificação já contidos na biblioteca da ferramenta *Weka*, utilizando a classe *NaiveBayesMultinomial* do pacote *weka.classifiers.bayes*.

Para implementação dos algoritmos *kNN Density-based* e *kNN Relative-Density-based* foi criada uma classe com os métodos que são pertinentes nas duas implementações, a *KNN_BasedSupervisedOneClass*. Nela foram desenvolvidos métodos para calcular a similaridade entre as instâncias utilizando medida de similaridade do cosseno, definir os vizinhos mais próximos de uma instância de teste, definir um valor de k vizinhos e também um método para somar a densidade de um determinado número de vizinhos. Para acelerar o cálculo de similaridade Cosseno foi utilizado uma estrutura do tipo *HashMap*.

Além de implementar a adaptação proposta na Seção 2.2.4, na implementação *IMBHN-based* foram desenvolvidos métodos para definição do erro quadrático mínimo, definição da taxa de correção de erros e também para o número máximo de iterações.

Para desenvolver o algoritmo *kMeans-based* foi criada uma classe para implementar métodos comuns a todos os algoritmos baseados em classificação utilizando *clusters*, tais como *kMedoids* e o próprio *kMeans*. O nome atribuído à esta classe foi *PrototypeBasedClusteringClassifier*, e ela contém métodos para definir os k grupos, definir número máximo de iterações, definir taxa de troca de grupos para as instâncias, definir o valor de convergência do algoritmo e também um método para atribuir um número de tentativas de inicialização dos centróides. Vale ressaltar que neste trabalho só foi utilizado o algoritmo *kMeans*.

Avaliação

Para obter as performances de classificação dos algoritmos e com isso poder compará-los, foi utilizada primeiramente a estratégia de validação cruzada, conhecida também

como *x-fold cross validation*. É uma das técnicas mais utilizadas para avaliar se os classificadores estão obtendo resultados significativos para o usuário [Tan et al. 2006, Kohavi et al. 1995]. Nesta abordagem, os exemplos fornecidos para treino são divididos em *x* pastas, onde $x-1$ pastas são aplicadas para treinamento do modelo de classificação, e o restante é aplicado para testar se o classificador está classificando documentos não rotulados corretamente. A partir desse momento, as pastas vão alternando de função, de modo que todas sejam utilizadas em algum momento tanto como treino, quanto como teste.

A validação cruzada aplicada na classificação multi-classe é feita levando em consideração todo o conjunto de exemplos rotulados apresentados pelo usuário, dividindo o conjunto de acordo com a quantidade *x* de pastas definidas pelo usuário. No entanto, para classificação baseada em uma única classe, foi necessária uma adaptação desta abordagem. Neste caso, a validação cruzada é aplicada para cada classe, onde o conjunto de treino possui apenas exemplos da classe de interesse, e o conjunto de testes possui exemplos tanto da classe de interesse quanto de *outliers*. O algoritmo implementado para a avaliação experimental executada neste projeto está descrito no Algoritmo 1. Vale ressaltar que a implementação deste algoritmo foi feita de maneira paralelizada, visto que a adaptação realizada para classificação baseada em uma única classe torna a validação cruzada mais custosa computacionalmente do que a abordagem convencional.

Após executar o algoritmo em questão, é possível obter a performance de classificação. Para isso, obtem-se os resultados de classificação vindos da validação cruzada e gera-se uma matriz de confusão para analisar o número de acertos dos classificadores em relação as coleções textuais utilizadas [Tan et al. 2006]. O tamanho da matriz de confusão é proporcional ao número de rótulos possíveis de serem aplicados a um novo documento. No caso deste trabalho, a matriz de confusão será 2x2 tendo em vista a possibilidade de serem atribuídos apenas rótulos positivos (pertencentes a classe de interesse) ou negativos (pertencentes aos documentos *outliers*). Para construí-la e consequentemente obter a performance de classificação, é necessário atribuir os documentos rotulados para as seguintes categorias:

- **Verdadeiro Positivo (VP):** documentos pertencentes a classe de interesse que foram corretamente rotulados pelo classificador;
- **Falso Positivo (FP):** documentos *outliers* que foram rotulados como sendo da classe de interesse;
- **Falso Negativo (FN):** documentos pertencentes a classe de interesse que foram rotulados como *outliers*;
- **Verdadeiro Negativo (VN):** documentos *outliers* que foram corretamente rotulados como *outliers*.

A estrutura da matriz de confusão pode ser visualizada na Tabela 2. A partir desta, extrair medidas para avaliar a performance dos classificadores. As métricas utilizadas para avaliar performance neste presente trabalho e suas respectivas fórmulas estão listadas abaixo.

- **Precisão:** avalia quantos exemplos da classe de interesse o classificador acertou em relação a todos os documentos que foram preditos como sendo da classe de interesse.

Algoritmo 1: Validação cruzada "*One-Class Learning*"

Input: NumFolds `/* Número de folds */`
D: coleção de documentos
Output: Performances
`/* Lista de performances de classificação obtidas em cada passo do procedimento de valicação cruzada para o aprendizado baseado em uma única classe */`
begin
  Performance = $\emptyset$
  foreach $c_i \in \mathcal{C}$ *da coleção D* **do**
    for $i \leftarrow 1$ **to** *NumFolds* **do**
      ConjuntoTreino ← ConjuntoValidacaoCruzadaTreino(i,c_i)
      `/* Gera o conjunto de treino da i-ésima iteração do procedimento de validação cruzada considerando somente de documentos da classe` c_i `*/`
      ConjuntoTeste ← ConjuntoValidacaoCruzadaTeste(i,c_i)
      `/* Gera o conjunto de teste da i-ésima iteração do procedimento de validação cruzada considerando uma pasta de documentos da classe` c_i `e documentos dos conjuntos` $c_j, c_j \not\subset c_i$ `*/`
      ModeloClassificacao ← ConstruirClassificador(ConjuntoTreino)
      Performance ← Performance ∪
      CalculaPerformance(ModeloClassificacao, ConjuntoTeste)
    end
  end
end

$$Precisao = \frac{VP}{VP + FP} \tag{9}$$

- **Revocação:** avalia quantos exemplos da classe de interesse o classificar acertou em relação a todos os documentos que são de fato da classe de interesse.

$$Revocacao = \frac{VP}{VP + FN} \tag{10}$$

- **F_1:** é a média harmônica entre a precisão e a revocação, combinando ambas em uma única medida.

$$F_1 = \frac{2 \cdot Precisao \cdot Revocacao}{Precisao + Revocacao} \tag{11}$$

A medida F_1 utilizada no presente trabalho para fins de comparação de desempenho entre os algoritmos. Foi utilizado o maior valor de F_1 obtido pelos parâmetros de

Tabela 2. Matriz de confusão.

		Classe Real	
		Classe de Interesse (+)	Outlier (-)
Classe Predita	Classe de Interesse (+)	VP	FP
	Outlier (-)	FN	VN

cada algoritmo para realizar comparação entre os mesmos. Desta forma, é extraído o melhor valor de F_1 para todos os algoritmos em cada base de dados, a fim de poder comparar o desempenho apenas se baseando nos melhores índices de cada algoritmo.

Para execução as avaliações apresentadas, foi utilizada a classe *Evaluations*, já existente na ferramenta *TextCategorizationTool*. Esta classe centraliza todas as avaliações dos aprendizados de máquina utilizados na aplicação, tornando-se então necessária a implementação do método *SupervisedInductiveEvaluationOneClass* para avaliar o aprendizado supervisionado baseado em uma única classe. Sendo assim, são passados como parâmetros o classificador a ser avaliado, o conjunto de testes, o objeto da matriz de confusão a ser gerada, e o índice da classe de interesse no conjunto de testes. O resultado desta função é o preenchimento da matriz de confusão no padrão apresentado na Tabela 2.

Após o desenvolvimento do método *SupervisedInductiveEvaluationOneClass*, foi necessário desenvolver uma classe para obter resultados dos algoritmos de *One Class Classification*. A classe implementada foi a *ResultsOneClass*, onde é possível obter a precisão, revocação, F_1 e outras métricas a partir da matriz de confusão criada na classe *Evaluations*. Sendo assim, para instanciar a classe de resultados, é necessário passar como parâmetro do objeto o caminho do arquivo no qual se deseja escrever os resultados, o número de classes utilizadas na avaliação, o número de repetições e o número de pastas utilizadas da validação cruzada. A partir disso, é gerada uma matriz de três dimensões para cada métrica, onde a primeira dimensão corresponde as classes de teste, a segunda corresponde ao número de repetições e a terceira ao número de pastas. Ao preencher todos os valores das matrizes qe armazenam as performances de classificação, os resultados são gravados em um arquivo texto para posterior análise.

Avaliação Experimental

Uma vez descrito como foi desenvolvido o projeto de implementação, podemos definir então como serão realizadas as avaliações experimentais, definindo inicialmente como quais as coleções textuais serão utilizadas, depois apresentando as configurações estabelecidas por algoritmo, e por fim mostrando os resultados gerados a partir disso.

Coleções textuais

Para realizar a avaliação experimental foram utilizadas 25 coleções textuais extraídas de um *benchmarking* com coleções textuais de diversos domínios, apresentado em [Rossi et al. 2013]. É possível observar entre as coleções textuais escolhidas a presença de documentos dos seguintes domínios: páginas *web* (WP), análise de sentimentos (SA), ar-

tigos de notícias (NA), documentos médicos (MD), documentos científicos (TD), e-mails (EM) e resumos (SD).

Na Tabela 3 são apresentadas características a respeito das coleções textuais que serão utilizadas no presente trabalho. Para cada coleção textual são apresentadas as seguintes características:

- $|\mathcal{D}|$: número de documentos contidos na coleção textual;
- $|\mathcal{T}|$**:** número de termos contidos nos documentos da coleção textual;
- $\overline{|\mathcal{T}|}$**:** média de ocorrência de termos nos documentos da coleção textual;
- $|\mathcal{C}|$**:** número de classes;
- $\sigma(\mathcal{C})$**:** desvio padrão considerando o percentual de ocorrência das classes;
- $max(\mathcal{C})$**:** porcentagem de documentos pertencentes à classe majoritária.

Tabela 3. Características de todas as coleções textuais analisadas.

Coleção	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{T}\rvert$	$\overline{\lvert\mathcal{T}\rvert}$	M.S.	$\lvert\mathcal{C}\rvert$	$\sigma(\mathcal{C})$	$max(\mathcal{C})$	**Domínio**
Classic4	7095	7749	35.28	99.54%	4	13.70%	45.16%	SD
CSTR	299	1726	54.27	96.86%	4	15.89%	42.81%	SD
Dmoz-Health-500	6500	4217	12.39	99.71%	13	0.00%	7.69%	WP
FBIS	2463	2001	159.24	92.04%	17	5.66%	20.54%	NA
La1s	3204	13196	144.63	98.90%	6	8.22%	29.43%	NA
La2s	3075	12433	144.82	98.84%	6	8.59%	29.43%	NA
Multi-Domain-Sentiment	8000	13360	42.36	99.68%	2	0.00%	50.00%	SA
NSF	10524	3888	6.55	99.83%	16	3.82%	13.39%	SD
Oh0	1003	3183	52.50	98.35%	10	5.33%	19.34%	MD
Oh10	1050	3239	55.63	98.28%	10	4.25%	15.71%	MD
Oh5	918	3013	54.43	98.19%	10	3.72%	16.23%	MD
Opinosis	6457	2693	7.55	99.72%	51	1.42%	8.18%	SA
Review Polarity	2000	15698	205.06	98.69%	2	0.00%	50.00%	SA
Re0	1504	2887	51.72	98.21%	13	11.56%	40.43%	NA
Reviews	4069	22927	183.10	99.20%	5	12.80%	34.11%	NA
SyskillWebert	334	4340	93.15	97.85%	4	10.75%	41.02%	WP
Tr11	414	6430	281.66	95.62%	9	9.80%	31.88%	TD
Tr12	313	5805	273.59	95.29%	8	7.98%	29.71%	TD
Tr21	336	7903	469.86	94.05%	6	25.88%	68.75%	TD
Tr23	204	5833	385.29	93.39%	6	15.58%	44.61%	TD
Tr31	927	10129	268.49	97.35%	7	13.37%	37.97%	TD
Tr41	8778	7455	19.53	99.74%	10	0.91%	2.77%	TD
Tr45	690	8262	280.58	96.60%	10	6.69%	23.19%	TD
WAP	1560	8461	141.33	98.33%	20	5.20%	21.86%	WP
WebKB	8282	22892	89.77	99.61%	7	15.19%	45.45%	WP

Não foi preciso aplicar qualquer pré-processamento nas coleções textuais, uma vez que elas já foram obtidas no formato de matriz documento-termo, com as *stopwords* removidas e as palavras padronizadas e radicalizadas. Os termos correspondem a palavras simples e o peso dos termos nessas representações corresponde ao *term-frequency*.

Configuração experimental

Uma vez implementados os algoritmos e aplicados à ferramenta, é preciso definir os parâmetros dos algoritmos desenvolvidos. Os algoritmos e seus respectivos parâmetros utilizados nesta avaliação experimental são:

(a) *Multinomial Naive-Bayes*
- Sem parâmetros.

(b) *kNN Density*
- k = [1-30].

(c) *kNN Relative-Density*
- k = [1-30].

(d) *IMBHN*
- Taxa de correção de erros: 0.01, 0.05, 0.1, 0.5;
- Erro quadrático mínimo: 0.001, 0.005, 0.01, 0.05;
- Número máximo de iterações: 1000.

(e) *kMeans*
- Número de *k*s: 1 a 30;
- Número máximo de iterações: 1000;
- Porcentagem para troca de grupo: 0%;
- Diferença mínima da medida objetiva: 0.0;
- Número de tentativas: 10.

Para realizar validação cruzada, foi definido o valor de $x = 10$.

Tendo em vista a particularidade de cada algoritmo e também das coleções textuais utilizadas, é necessário aplicar uma variedade de *thresholds* para que seja possível extrair qual o melhor *threshold* para cada algoritmo em cada coleção. Sendo assim, os *thresholds* definidos foram $\{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95\}$.

Além dos *thresholds* definidos manualmente foi utilizada a métrica *Three Sigma*, onde é utilizada a distribuição normal para definir automaticamente os *thresholds* de acordo com a variação da base de dados utilizada [Breyfogle III 2003]. Os *thresholds* definidos pela métrica *Three Sigma* correspondem a média (μ) e a variação do desvio padrão (σ) de -3σ a +3σ a partir da média, totalizando oito *thresholds*.

Resultados

Uma vez atribuída a configuração experimental na ferramenta *Text Categorization Tool*, disponibilizando as coleções textuais definidas e executando os algoritmos desenvolvidos, foi possível obter a performance de classificação. Como já citado na Seção 4.3, para comparar o desempenho de cada algoritmo para cada coleção textual, foi utilizada a métrica F_1. Os resultados da avaliação experimental são apresentados na Tabela 4. Os resultados em negrito correspondem aos classificadores que tiveram melhor desempenho para a coleção textual em questão.

Baseando-se nos resultados, é possível avaliar que a alguns algoritmos podem se sair melhor em determinados domínios. Por exemplo, o algoritmo *kMeans* obteve

Tabela 4. Resultados da avaliação experimental.

Collections (Domain)	IMBHN^{R}	KME	KNN Density	KNN Rel. Density	MNB
Classic4 (Sci. Art.)	0.4894	**0.6121**	0.6002	0.5800	0.0789
CSTR (Sci. Art.)	0.5441	0.5216	**0.5978**	0.5491	0.0891
Dmoz_Health_500 (*Web* Page)	0.2993	0.4211	**0.4454**	0.4366	0.0170
Fbis (News)	**0.4304**	0.3790	0.3672	0.3587	0.0137
La1s (News)	0.1957	0.2709	**0.3485**	**0.3485**	0.0402
La2s (News)	0.1871	0.2930	**0.3551**	0.3549	0.0403
MultiDomainSent (Sent. Anal.)	0.1788	0.1814	0.1801	0.1797	**0.1833**
NSF (Sci. Art.)	0.3175	0.3321	**0.3937**	0.3724	0.0149
Oh0 (Med. Doc.)	**0.4290**	0.4080	0.3073	0.3154	0.0265
Oh10 (Med. Doc.)	0.3736	**0.3835**	0.3539	0.3428	0.0281
Oh5 (Med. Doc.)	0.3822	**0.4242**	0.3802	0.3767	0.0234
Opinosis (Sent. Anal.)	0.2440	**0.2589**	0.2587	0.2586	0.0042
ReviewPolarity (Sent. Anal.)	0.1772	0.1696	**0.2212**	**0.2212**	0.1667
Re0 (News)	0.1497	0.2866	**0.3102**	0.2984	0.0201
Reviews (News)	**0.4945**	0.4761	0.3090	0.3607	0.0517
SyskilllWebert (*Web* Page)	0.6135	**0.7662**	0.7483	0.7219	0.0710
Tr11 (Inf. Retriev.)	0.2699	**0.4459**	0.4216	0.4200	0.0268
Tr12 (Inf. Retriev.)	0.3519	**0.4718**	0.4331	0.3986	0.0307
Tr21 (Inf. Retriev.)	0.2486	**0.4254**	0.3103	0.3103	0.0771
Tr23 (Inf. Retriev.)	0.3041	**0.3791**	0.3567	0.3555	0.0462
Tr31 (Inf. Retriev.)	**0.4540**	0.4498	0.4021	0.4000	0.0412
Tr41 (Inf. Retriev.)	0.4717	0.5176	**0.5828**	0.5659	0.0237
Tr45 (Inf. Retriev.)	0.4491	**0.4972**	0.4711	0.4312	0.0241
Wap (*Web* Page)	0.0721	0.2788	0.2892	**0.2901**	0.0111
WebKB (*Web* Page)	0.0737	0.1173	0.2511	**0.2543**	0.0943

os melhores resultados para documentos científicos, enquanto que artigos de notícias e páginas *web* são melhores classificados por algoritmos variantes do *kNN*. Enquanto isso, resumos e análises de sentimentos não possuem um padrão para o melhor classificador.

Uma vez extraídos os resultados de classificação com os valores de F_1, os mesmos foram analisados para verificar se os resultados dos algoritmos são estatisticamente diferentes. Para realizar esta análise foi aplicado o Teste de Friedman com pós-teste de Nemenyi, um modelo de teste estatístico que busca verificar se há diferença estatisticamente significante entre os resultados obtidos por diferentes classificadores em diferentes *datasets*. [Trawinski et al. 2012, García et al. 2010, Demsar 2006].

Na Figura 6 é possível observar um diagrama com o resultado dos testes estatísticos realizados. No diagrama em questão, é apresentado o *ranking* médio dos algoritmos e aqueles conectados por uma linha, não apresentam diferenças estatisticamente significantes entre si.

Pode observar pela Figura 6 que o algoritmo de melhor desempenho médio foi do

kMeans e o de pior desempenho foi do *Multinomial Naive Bayes*. Entretanto, não houve diferenças estatisticamente significantes entre os resultados dos algoritmos.

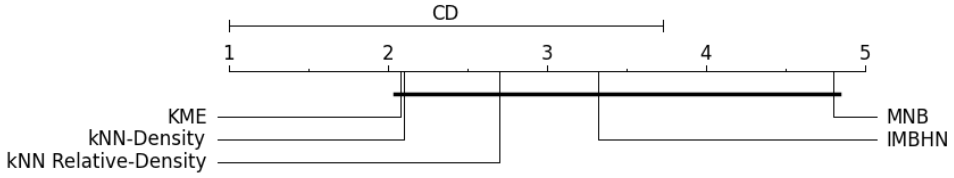


Figura 6. Diagrama do pós-teste de Nemenyi.

Conclusões e trabalhos futuros

A classificação automática de textos se mostra importante nos dias atuais dado o grande volume de textos produzidos, dificultando o processo de classificação manual dos mesmos. Entretanto, para tornar a aplicação da classificação automática de textos mais prática, áreas de aprendizado de máquina como o aprendizado baseado em uma única classe tem ganhado destaque nos últimos anos.

Entretanto, várias pesquisas na área de aprendizado baseado em uma única classe limitam-se à utilização de poucos algoritmos e/ou poucas coleções textuais em suas avaliações experimentais. Dado isso, neste projeto de conclusão de curso foram analisados os principais algoritmos da área de aprendizado supervisionado baseado em uma única classe em uma extensa avaliação experimental.

Com os resultados obtidos neste trabalho pode-se verificar que alguns algoritmos tiveram melhores resultados para alguns domínios específicos, porém foi possível verificar através de testes estatísticos que não houve diferença estatisticamente significante entre os classificadores.

Em trabalhos futuros pretende-se explorar o agrupamento das instâncias da classe de interesse antes da indução de alguns dos modelos de classificação, como o *IMBHN* e o *NB*, uma vez que os pontos da classe de interesse podem estar divididos em diferentes grupos no espaço. Sendo assim a indução de um único modelo considerando todos esses grupos de pontos dispersos podem gerar um modelo de classificação de menor qualidade.

Outra etapa a ser explorada é o uso de *ensemble* de classificadores, a fim de obter uma melhor performance em relação aos resultados apresentados neste trabalho. Por fim, pretende-se explorar o uso de aprendizado semissupervisionado baseado em uma única classe para melhorar as performances de classificação, uma vez que dados não rotulados são fáceis de coletar e podem ser úteis para gerar modelo de classificação de melhor qualidade.

Referências

Aggarwal, C. C. and Zhai, C. (2012). *Mining text data*. Springer Science & Business Media.

Bellinger, C., Sharma, S., Zaane, O. R., and Japkowicz, N. (2017). Sampling a longer life: Binary versus one-class classification revisited. In *First International Workshop on Learning with Imbalanced Domains: Theory and Applications*, pages 64–78.

Breyfogle III, F. W. (2003). *Implementing six sigma: smarter solutions using statistical methods*. John Wiley & Sons.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30.

Dias, M. A. L. and de Gomensoro Malheiros, M. (2005). Extração automática de palavras-chave de textos da língua portuguesa. *Centro Universitário UNIVATES*.

García, S., Fernández, A., Luengo, J., and Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044–2064.

Gosling, J. (2000). *The Java language specification*. Addison-Wesley Professional.

Gosling, J. and McGilton, H. (1995). The java language environment. *Sun Microsystems Computer Company*, 2550.

Hotho, A., Staab, S., and Stumme, G. (2003). Ontologies improve text document clustering. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 541–544. IEEE.

Kemmler, M., Rodner, E., Wacker, E.-S., and Denzler, J. (2013). One-class classification with gaussian processes. *Pattern Recognition*, 46(12):3507–3518.

Khan, S. S. and Madden, M. G. (2009). A survey of recent trends in one class classification. In *Irish Conference on Artificial Intelligence and Cognitive Science*, pages 188–197. Springer.

Kohavi, R. et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Stanford, CA.

Kumar, B. S. and Ravi, V. (2017). One-class text document classification with ocsvm and lsi. In *Artificial Intelligence and Evolutionary Computations in Engineering Systems*, pages 597–606. Springer.

Li, W., Guo, Q., and Elkan, C. (2011). A positive and unlabeled learning algorithm for one-class classification of remote-sensing data. *IEEE Transactions on Geoscience and Remote Sensing*, 49(2):717–725.

Liu, B., Dai, Y., Li, X., Lee, W. S., and Yu, P. S. (2003). Building text classifiers using positive and unlabeled examples. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 179–186. IEEE.

Liu, X., Liu, H., Gong, H., Lin, Z., and Lv, S. (2017). Appling the one-class classification method of maxent to detect an invasive plant spartina alterniflora with time-series analysis. *Remote Sensing*, 9(11):1120.

Manevitz, L. M. and Yousef, M. (2001). One-class svms for document classification. *Journal of Machine Learning Research*, 2(Dec):139–154.

Peng, T., Zuo, W., and He, F. (2006). Text classification from positive and unlabeled documents based on ga. In *Proc. of VECPAR*, volume 6.

Rossi, R. G. (2016). *Classificação automática de textos por meio de aprendizado de máquina baseado em redes*. PhD thesis, Universidade de São Paulo.

Rossi, R. G., de Andrade Lopes, A., and Rezende, S. O. (2016). Optimization and label propagation in bipartite heterogeneous networks to improve transductive classification of texts. *Information Processing & Management*, 52(2):217–257.

Rossi, R. G., de Paulo Faleiros, T., de Andrade Lopes, A., and Rezende, S. O. (2012). Inductive model generation for text categorization using a bipartite heterogeneous network. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 1086–1091. IEEE.

Rossi, R. G., Marcacini, R. M., and Rezende, S. O. (2013). Benchmarking text collections for classification and clustering tasks. *Institute of Mathematics and Computer Sciences, University of Sao Paulo*.

Salazar, F. A. S. R. (2012). *Um estudo sobre o papel de medidas de similaridade em visualização de coleções de documentos*. PhD thesis, Universidade de São Paulo.

Salton, G. (1989). Automatic text processing: The transformation, analysis, and retrieval of. *Reading: Addison-Wesley*.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Sun, Y. and Han, J. (2012). Mining heterogeneous information networks: principles and methodologies. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 3(2):1–159.

Tan, P.-N. et al. (2006). *Introduction to data mining*. Pearson Education India.

Tax, D. M. J. (2001). One-class classification.

Trawinski, B., Smetek, M., Telec, Z., and Lasota, T. (2012). Nonparametric statistical analysis for multiple comparison of machine learning regression algorithms. *Applied Mathematics and Computer Science*, 22(4):867–881.

Turner, V., Gantz, J. F., Reinsel, D., and Minton, S. (2014). The digital universe of opportunities: Rich data and the increasing value of the internet of things.

Weiss, S. M., Indurkhya, N., and Zhang, T. (2010). *Fundamentals of predictive text mining*, volume 41. Springer.

Wilbur, W. J. and Sirotkin, K. (1992). The automatic identification of stop words. *Journal of information science*, 18(1):45–55.

Witten, I. H., Frank, E., and Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition.

Zhuang, L. and Dai, H. (2006). Parameter estimation of one-class svm on imbalance text classification. *Advances in Artificial Intelligence*, pages 538–549.